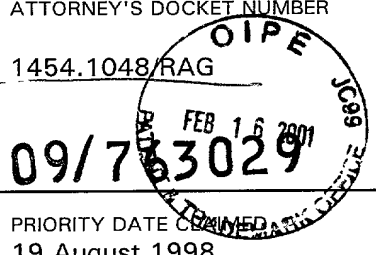


FORM PTO-1390 U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE		ATTORNEY'S DOCKET NUMBER 1454.1048/RAG
TRANSMITTAL LETTER TO THE UNITED STATES DESIGNATED/ELECTED OFFICE (DO/EO/US) CONCERNING A FILING UNDER 35 U.S.C. 371		
INTERNATIONAL APPLICATION NO. PCT/DE99/02013	INTERNATIONAL FILING DATE 1 July 1999	PRIORITY DATE CLAIMED 19 August 1998
TITLE OF INVENTION METHOD ARRAY AND SET OF SEVERAL ARRAYS FOR PROTECTING SEVERAL PROGRAMS AND/OR FILES FROM UNAUTHORIZED ACCESS BY A PROCESS		
APPLICANT(S) FOR DO/EO/US Manfred SCHAEFER		
Applicant herewith submits to the United States Designated/Elected Office (DO/EO/US) the following items and other information:		
<ol style="list-style-type: none"> 1. <input checked="" type="checkbox"/> This is a FIRST submission of items concerning a filing under 35 U.S.C. 371. 2. <input checked="" type="checkbox"/> This is an express request to immediately begin national examination procedures (35 U.S.C. 371(f)). 3. <input type="checkbox"/> The US has been elected by the expiration of 19 months from the priority date (PCT Article 31). 4. <input checked="" type="checkbox"/> A copy of the International Application as filed (35 U.S.C. 371(c)(2)) <ol style="list-style-type: none"> a. <input checked="" type="checkbox"/> is transmitted herewith (required only if not transmitted by the International Bureau). b. <input type="checkbox"/> has been transmitted by the International Bureau. c. <input type="checkbox"/> is not required, as the application was filed in the United States Receiving Office (RO/US). 5. <input type="checkbox"/> A translation of the International Application into English (35 U.S.C. 371(c)(2)). 6. <input type="checkbox"/> Amendments to the claims of the International Application under PCT Article 19 (35 U.S.C. 371(c)(3)) <ol style="list-style-type: none"> a. <input type="checkbox"/> are transmitted herewith (required only if not transmitted by the International Bureau). b. <input type="checkbox"/> have been transmitted by the International Bureau. c. <input type="checkbox"/> is not required, as the application was filed in the United States Receiving Office (RO/US). 7. <input type="checkbox"/> A translation of the amendments to the claims under PCT Article 19 (35 U.S.C. 371(c)(3)). 8. <input type="checkbox"/> An oath or declaration of the inventor (35 U.S.C. 371(c)(4)). 9. <input type="checkbox"/> A translation of the Annexes to the International Preliminary Examination Report under PCT Article 36 (35 U.S.C. 371(c)(5)). 		
Items 10-15 below concern document(s) or information included:		
<ol style="list-style-type: none"> 10. <input type="checkbox"/> An Information Disclosure Statement Under 37 CFR 1.97 and 1.98. 11. <input type="checkbox"/> An assignment document for recording. Please mail the recorded assignment document to: <ol style="list-style-type: none"> a. <input type="checkbox"/> the person whose signature, name & address appears at the bottom of this document. b. <input type="checkbox"/> the following: 12. <input type="checkbox"/> A preliminary amendment. 13. <input type="checkbox"/> A substitute specification 14. <input type="checkbox"/> A change of power of attorney and/or address letter. 15. <input checked="" type="checkbox"/> Other items or information: International Preliminary Examination Report with amended claims and International Search Report 		

2. [X] The U.S. National Fee (35 U.S.C. 371(c)(1)) and other fees as follows:					
CLAIMS	(1) FOR	(2) NUMBER FILED	(3) NUMBER EXTRA	(4) RATE	(5) CALCULATIONS
	TOTAL CLAIMS	22 -20 =	2	x \$ 18.00	36.00
	INDEPENDENT CLAIMS	5 -3 =	2	x \$ 80.00	160.00
	MULTIPLE DEPENDENT CLAIM(S) (if applicable)			+ \$270.00	0.00
	BASIC NATIONAL FEE (37 CFR 1.492(a)(1)-(4): <input type="checkbox"/> Neither international preliminary examination fee (37 CFR 1.482) nor international search fee (37 CFR 1.445(a)(2)) paid to USPTO.....\$1,000 <input checked="" type="checkbox"/> International preliminary examination fee (37 C.F.R. 1.482) not paid to USPTO but International Search Report prepared by the EPO or JPO.. . . . \$ 860 <input type="checkbox"/> International preliminary examination fee (37 C.F.R. 1.482) not paid to USPTO but international search fee (37 C.F.R. 1.445(a)(2)) paid to USPTO.....\$ 710 <input type="checkbox"/> International preliminary examination fee paid to USPTO (37 CFR 1.482) but all claims did not satisfy provision of PCT Article 33(1)-(4).....\$ 690 <input type="checkbox"/> International preliminary examination fee paid to USPTO (37 CFR 1.482) and all claims satisfied provisions of PCT Article 33(2) to (4).....\$ 100				860.00
	Surcharge of \$130 for furnishing the National fee or oath or declaration later than <input type="checkbox"/> 20 <input type="checkbox"/> 30 mos. from the earliest claimed priority date (37 CFR 1.482(e)).				0.00
	TOTAL OF ABOVE CALCULATIONS				1,056.00
	Reduction by 1/2 for filing by small entity, if applicable. Affidavit must be filed also. (Note 37 CFR 1.9, 1.27, 1.28.)				
	SUBTOTAL				1,056.00
	Processing fee of \$130 for furnishing the English Translation later than <input type="checkbox"/> 20 <input type="checkbox"/> 30 mos. from the earliest claimed priority date (37 CFR 1.482(f)).				.00
	TOTAL NATIONAL FEE				1,056.00
	Fee for recording the enclosed assignment (37 CFR 1.21(h)).				+
	TOTAL FEES ENCLOSED				1,056.00

- a. [X] A check in the amount of \$1,056.00 to cover the above fees is enclosed.
- b. ☐ Please charge my Deposit Account No. 19-3935 in the Amount of \$ to cover the above fees. A duplicate copy of this sheet is enclosed.
- c. [X] The Commissioner is hereby authorized to charge any additional fees which may be required, or credit any overpayment to Deposit Account No. 19-3935. A duplicate copy of this sheet is enclosed.



21171

PATENT TRADEMARK OFFICE

2/16/01

DATE

Richard A. Gollhofer

Richard A. Gollhofer
REGISTRATION NO. 31,106

Description

Method, array and a set of several arrays for protecting several programs and/or files from unauthorized access by a process

The invention pertains to a method, an array and a set of several arrays for protecting several programs from unauthorized access by a process.

A method and an array for protecting several programs from unauthorized access by a user is known from [1] or [6]. In the method according to [1], the access protection for a program is realized in that an access authorization file is assigned to each user of a system. If a process attempts to access a program, a check is made to confirm whether the user who started the process has the right to access the program in question. The access is allowed only if the process was started by an authorized user, who is thus equipped with the access rights.

Known from [2] is a so-called virus scanner. A virus scanner checks the stored, known sequence of data by means of which the program is realized. If a discrepancy from the known sequence is found, a message is sent to a user of the system that there is a possibility that the system contains a virus.

Also known from [1] is an operating system for a computer. The operating system known from [1] exhibits a variety of security gaps through which a hacker can endanger the integrity of programs that are executed by using the operating system.

A possible mechanism for endangering the protection of programs during the use of an operating system is also described in [5].

In [7], a computer system for the licensing of software is described.

The invention is thus based on the problem of protecting several programs and/or several files from unauthorized access by a process when using an operating system that exhibits fundamental security gaps.

The problem is solved by the method and by the array in accordance with the features of the independent patent claims.

In a method for protecting several programs and/or several files from unauthorized access by a process, each program and/or each file to be protected is assigned an address space. In addition, each program and/or each file to be protected is also assigned a process file, whereby the process or processes that may run in the address space in question is or are stored in a process file. During the running of a program and/or a file to be protected, for a process that attempts to access the address space of the program and/or the file to be protected, a check is made to confirm whether the accessing process is included in the corresponding process file. If the accessing process is included in the process file, the accessing process is started; otherwise the accessing process is not started.

In an additional method for protecting several programs and/or several files from unauthorized

access by a process, each program and/or each file to be protected is assigned an address space. In addition, each program and/or file to be protected is also assigned a process file, whereby the process or processes that may run in the address space in question is or are stored in a process file. During the running of a program and/or a file to be protected, for a process that attempts to access the address space of the program and/or a file to be protected, a check is made to confirm whether the accessing process is included in the corresponding process file. If the accessing process is included in the process file, the accessing process is continued; otherwise the accessing process is terminated.

An array for protecting several programs from unauthorized access by a process exhibits a processor that is set up in a way such that the following steps can be carried out:

- an address space is assigned to each program and/or each file to be protected,
- a process file is assigned to each program and/or each file to be protected,
- the process or processes that may run in the address space in question is or are stored in a process file,
- during the running of a program and/or a file to be protected, for a process that attempts to access the address space of the program and/or the file to be protected, a check is made to confirm whether the accessing process is included in the corresponding process file,
- if the accessing process is included in the process file, the accessing process is started, and
- otherwise the accessing process is not started.

A further array for protecting several programs from unauthorized access by a process exhibits a

processor that is set up in a way such that the following steps can be carried out:

- an address space is assigned to each program and/or each file to be protected,
- a process file is assigned to each program and/or each file to be protected,
- the process or processes that may run in the address space in question is or are stored in a process file,
- during the running of a program and/or a file to be protected, for a process that attempts to access the address space of the program and/or the file to be protected, a check is made to confirm whether the accessing process is included in the corresponding process file,
- if the accessing process is included in the process file, the accessing process is continued, and
- otherwise the accessing process is terminated.

A set of several arrays and a server array which is connected with each array of the set of several arrays and which is to protect several programs from unauthorized access by a process are set up in a way such that each array exhibits a processor that is set up in such a way that the following steps can be carried out:

- an address space is assigned to each program and/or each file to be protected,
- a process file is assigned to each program and/or each file to be protected,
- the process or processes that may run in the address space in question is or are stored in a process file,
- during the running of a program and/or a file to be protected, for a process that attempts to access the address space of the program and/or the file to be protected, a check is made to confirm whether

Foreign Version

the accessing process is included in the corresponding process file,

- if the accessing process is included in the process file, the accessing process is started or continued, and

- otherwise an alarm signal is generated and sent to the server array,

and the server array exhibits a processor that is set up in such a way that a predetermined action is triggered in dependency on at least one received alarm signal.

Address space in this context is understood to mean a program area has one program assigned to it.

The invention closes several security gaps of the operating system described in [1].

Moreover, the particular program to be protected is protected against a procedural attack (attack on a process), e.g., against a Trojan horse.

In addition, a significant advantage of the invention is seen in the fact that with scalable effort, a defined degree of security can be assured for the program to be protected.

Through the set of several arrays, each of which is connected to the server array, local protection at the arrays is possible in such a way that in the event an attack is detected, an alarm signal is generated and sent to the server array in which a predetermined action is carried out centrally. That way, it is possible to discover local processes that are not known to the server array itself.

Preferred further developments of the invention arise from the dependent claims.

In a further development, to increase the level of security that can be achieved it is advantageous to form for at least a part of the processes included in a process file a unique identifying cryptographic value, whereby the value in question is contained in the process file. The accessing process's cryptographic value is formed, and the cryptographic values of the processes are compared with each other during the check.

The cryptographic value can be a digital signature. However, a general one-way function can also be formed by using a hash function.

In a further configuration, it is advantageous to forward, in a call mechanism for a function of an operating system core with which the programs are executed, a call of the accessing process to a checking function in which the check is carried out. An efficient and thus cost-effective realization of the invention is possible in this way.

The checking function can be integrated into the monitored address space as a dynamically integrated file, which results in the achieving of a further improvement in the protective effect.

A call of an accessing process can also be forwarded to a checking function that is integrated into the operating system core, whereby the check takes place in the checking function. The security of the protection can be further increased for the programs in this way.

A further increase in the achievable level of security can be assured if a protection program that is set up in such a way that the invention can be put into practice is stored encoded and is decoded at the start of the method. After the protection program has been decoded, its integrity can be checked,

and the method is carried out only if the integrity of the protection program is assured. After the integrity test of the protection program, the integrity of all of the processes contained in the process files can be checked, and the method is carried out only if the integrity of the protection program is assured. After the integrity test of the processes, the integrity of the program to be protected can be checked, and the program should be executed only if the integrity of the protection program is assured.

The invention can be used advantageously in the operating system described in [1].

Although the embodiment that is explained below describes the protection of programs, protection of several files is also easily possible using the same procedure.

An embodiment of the invention is shown in the Figures and will be described in more detail below.

The following are shown:

Figure 1, a sketch in which the principle on which the invention is based is shown symbolically;

Figure 2, a sketch in which a process layer model is shown;

Figure 3, a block diagram in which a computer network is shown;

Figure 4, a flow chart in which the individual steps of the method of the embodiment are shown;

Foreign Version

Figure 5, a sketch in which the principle of a possible integration of the invention into an operating system is shown;

Figure 6, a sketch in which the possible realization according to Figure 5 is shown in detail;

Figure 7, a sketch in which an additional possible integration of the invention into an operating system is shown.

Fig. 3 shows a first computer 301 with an input/output interface 302 that is connected via a bus 303 with a memory 304 and a processor 305. Through the input/output interface 302, the first computer 301 is connected via a computer network 306 with a multiplicity of computers 307, 308, 309, 310, 311.

The communications protocol used to transmit digital data is the TCP/IP protocol (Transmission Control Protocol/Internet Protocol).

The invention protects the first computer 301 against unauthorized accesses by processes that are either stored in the memory 304 of the first computer 301 or access/affect the first computer 301 from the other computers 307, 308, 309, 310, 311.

The operating system described in [1] is implemented in the first computer 301.

The principle that is explained below and that forms the basis for the method and the array is shown graphically in **Fig. 1**.

Symbolically represented programs 101, 102, 103 are to be protected by the invention against unauthorized access by at least one process 104, 105, 106 that attempts or attempt to access a program 101, 102, 103.

The programs 101, 102, 103 and processes 104, 105, 106 use as the operating system the operating system described in [1], which is represented as a unit 107 that surrounds the programs 101, 102, 103 and processes 104, 105, 106.

A program-specific “protective covering” 108, 109, 110 is graphically formed by the method or the array around each of the programs 101, 102, 103 to be protected. Through the program-specific safeguarding, a freely scalable level of security is achieved for the program 101, 102, 103 to be protected.

As is shown in a process layer model 201 in **Fig. 2**, the method or array can be realized on various logical levels of the programs 101, 102, 103 to be protected. **Fig. 2** shows three logical levels in the process layer model 201.

Safeguarding against a hacker 205 can take place at the level of the application programs 202 to be protected, at the level of the operating system programs 203 to be protected, or at the level of the operating system core, as well as on the system hardware 207.

The closer to the system hardware 207 the safeguarding of a program takes place, the greater is the level of security that is achieved through the invention.

The protective covering 206 is “placed” around the program at the time the program 101, 102, 103 to be protected is being run.

The method is realized in the form of cyclical, concurrent processes. The method is explained with the aid of the flow chart shown in **Fig. 4**.

Foreign Version

The first thing done following the start of the operating system (step 401) is the start of a protection program (step 402). The protection program is set up in such a way that the method described below can be executed. The protection program is stored in encoded form, as a result of which any change to the protection program itself is impossible.

The encoding of the protection program also prevents a detailed analysis of the program that is executing the method.

When the protection program is started (step 402), the protection program is decoded by a predetermined start routine, which contains the key needed for decoding the protection program and possibly additional basic functions of the operating system as well, which results in the determination of the actual program code of the protection program.

In this way, the protection program is protected in the active state against off-line attacks such as disassembling, debugging, patching, etc.

After the decoding (step 402) of the protection program, the integrity of the protection program is checked dynamically (step 403).

If the integrity of the protection program is not assured, the method is terminated (step 404).

In a further step, the integrity of the operating system processes is checked dynamically (step 405).

If the integrity test is negative, the method is again terminated (step 404).

If the integrity of the processes of the operating system 405 is assured, the program to be protected is started.

Foreign Version

The method described above is carried out dynamically for each program 101, 102, 103 to be protected.

One process file 111, 112, 113 is assigned to each program 101, 102, 103 to be protected.

The processes that may run in an address space that is also uniquely assigned to each program 101, 102, 103 are included in a process file 111, 112, 113 for the program 101, 102, 103 which is to be protected and to which the process file 111, 112, 113 is assigned. The data are stored in the process files by means of a hash function that uniquely identifies the process in question.

Following the start of the program 101, 102, 103 in question (step 407), the integrity of the program 101, 102, 103 itself is checked (step 408).

If the integrity test is negative, the method is again terminated (step 404).

If the integrity test is positive, i.e., if the integrity of the program 101, 102, 103 is assured, the method for the program 101, 102, 103 to be protected is repeated until the program to be protected is itself terminated (step 409).

The method is iterated in dependency on a predefinable event or at predefinable interval of time between two executions of the method (step 410).

As soon as a process 104, 105, 106 attempts to access the address space or the program 101, 102, 103 itself, in a further step (step 411) a check is made to whether or not the process 104, 105, 106 is contained in the process file of the program 101, 102, 103 to be protected which the process 104,

Foreign Version

105, 106 is trying to access.

This takes place through the formation of a hash value on the accessing process 104, 105, 106 and comparison of the hash value of the accessing process 104, 105, 106 with the hash values stored in the process file. If the accessing process is included in the process file of the program 101, 102, 103 to be protected, then the process 104, 105, 106 is executed (step 412).

Otherwise, the process 104, 105, 106 is not started (step 413) and the user is informed of a possible attack on his program 101, 102, 103.

For each process that is active in the address space of a program 101, 102, 103, i.e., the process that is running, a check is made at predefinable intervals of time or in an event-controlled way to confirm whether the process in question is contained in the process file of the corresponding program 101, 102, 103 whose address space is being examined. If that is not the case, then the corresponding process is terminated and the user is notified of a possible attack on his program 101, 102, 103.

Regular monitoring of the program is assured in this way.

Options for the integration of the method described above into the operating system described in [3] are described below.

Option 1:

Integration of a dynamically linked file 501 into the application programming interface (Application

Programming Interface, API) (cf. **Fig. 5**).

Foreign Version

The dynamically linked file 501 becomes active in the address space of the potential accessing program 502. Using the dynamically linked file 501, the following steps are carried out in the address space of the potential accessing program 502:

- From the dynamically linked file 501, all of the identifiers (module handles) are determined for every additional dynamically linked file that contains interface calls to be monitored. By doing this, all accesses to all of the dynamically linked files to be monitored can be enabled.

Considered to be attack-relevant are all interface calls for the direct or indirect starting, ending and controlling of processes, e.g., write accesses to the process memory, changes to access rights, in principle, all interface instructions that work with external process identifiers (process handles), all instructions for the realization of message hooks (a programmable filtering function for messages for GUI (Graphic User Interface) intercommunication and for process intercommunication), and for debugging purposes.

The term “indirect starting” is also understood to mean compiler mechanisms for the automatic modification of program code, OLE (Object Link Embedding) and RPC mechanisms (Remote Procedure Call), as well as accesses from other operating system programming interfaces. This term additionally includes the control of the mechanisms that are used for the execution of dynamically linked files 501 (active X instructions, etc.).

- In addition, writing rights for the address space are assigned to the dynamically linked files 501.
- A branch instruction is stored in place of the original interface call, and the replaced instructions of the original code 503 are saved.

If a process 504 now attempts by means of an interface call `APIFktCall ()` to access the program 502 to be protected or to start an inactive program, the protection program 506 is called

via the dynamically linked file 501. As the transfer parameter for protection program 506, protection program 506 is told which program 502 is to be called by the accessing process 504.

Through a comparison with the allowable interface calls included in the process file of the program 502 in question, in accordance with the method described above, a decision can be made as to whether the interface call 505 is allowable or not.

The accessing process 504 is then either executed or “blocked.” If a decision is made in the protection program 506 that the accessing process 504 should be executed, the original code of the interface call 503 is executed and, following its execution, a return code 507 is sent back to the accessing process 504. Otherwise, an error message 508 is reported to the accessing process 504.

The method described above makes use of the basic mechanism of so-called DLL injection, which is known from [5].

Option 2:

Fig. 6 shows a refined realization of the principle described in **Fig. 5**. This version is especially well suited for a case in which a process that has been declared “safe” is mistakenly located in the address space that the dynamically linked file 601 itself accesses.

The method described in the following ensures that an access to predefined data is possible only from the dynamically linked file 601 itself, and accesses from a different instruction sequence segment, particularly that of an accessing application, will be prevented.

The method presented in the following is preferably realized with static modification of the method described above.

The following assumptions are made with regard to the method described below:

- Data to be protected are stored in a protected area 602 that is set up and described during the initialization of the dynamically linked file 601, and then receive a protection attribute so that area 602 can be accessed only through the dynamically linked file 601 itself.
- All areas that contain executable code receive the protection attribute “page_execute”, which prevents the executable code in question from being changed unless the protection attribute is changed beforehand.
- Every interface function 603 is safeguarded in the following way: An entry address for the interface function 603 is replaced by a modified entry address that leads to a modified interface function 604.

In the modified interface function, a branch is made to an interface process 605, which is contained in the dynamically linked file 601. This interface process branches to protection program 606, which, for a calling process 607 that attempts to access the interface function 603 with an interface call 608, checks if this call is allowed for the accessing process 607.

If it is, the interface function 603 is executed and after the execution of interface function 603, another branch is made to the modified interface function 604, which is symbolized by an arrow 609. Following the execution of additional predefinable instructions, a branch is made to an interface return function 610, which is indicated by an arrow 611. This takes place by means of a

branch instruction. In the interface return function 610, a check is made once again (step 612) to confirm whether the interface call 608 is allowed. If it is not, an error message 613 is sent to the process 607. This takes place in exactly the same way if, using the procedure described above, it was determined in interface function 605 that the interface call is not allowed.

However, if in the checking step 612 in the interface return function 610 it is also determined that the interface call is allowed, then the result of the called interface function is sent to the accessing process 607 (step 614).

Option 3:

The invention can also be integrated in the operating system core. This form of implementation makes it possible to integrate a control mechanism that can no longer be bypassed under user access rights, but rather, only under the access rights of the system administrator. This results in a substantial increase in the level of security.

An integration mechanism that can be used for this purpose is described in [4]. Under this mechanism, with the transfer into the mode of the operating system core (kernel mode), interface calls are “intercepted” in the operating system core itself. In this instance, the administrative mechanism for the interrupt routine is realized in the operating system core, and is therefore protected against accesses by processes that are active in the user mode. A summary of a number of such alternative implementation options can be found in [4].

Fig. 7 shows an overview of two realization options described above.

Foreign Version

An application program 701 (application) uses functions of the operating system for the program run.

The operating system functions are grouped into functions 702 of the operating system in a user mode, and functions 703 of the kernel mode. Both of these functions are symbolically illustrated as blocks.

Through the use of dynamically linked files (*.dll) it is possible to integrate the method within the framework of the user mode, which can take place through a first block 704 through the use of a dynamically linked file "NTDLL.DLL" 705 of the operating system described in [1].

The other option for integrating the method into the operating system core is indicated by a second block 706, whereby in this version of the integration, the mechanism is integrated during the transition from the user mode into the kernel mode.

Several alternatives to the implementation examples explained above will be presented in the following:

The protection of the protection program can be increased by integrating the start routine in the operating system core, e.g., as a so-called kernel mode process or as system service.

The dynamically linked file can also be provided both statically and dynamically, i.e., only during the run time of the program 502 to be protected, or during the entire run time of the operating system.

Alternative software interrupt routines can also be used as alternatives to the dynamically linked

[illegible]

The invention can be realized both by means of software as well as by means of hardware, or partly by means of software and partly by means of hardware.

In addition, as is shown in **Fig. 4**, if the integrity test is negative (step 408), a check is made in an additional checking step (step 414) to confirm whether a reloading of the original protection program and/or a program to be protected is possible from a trustworthy authority, or if a recovery of the protection program and/or the program to be protected is possible in a way such that its/their integrity is assured.

If that is not possible, the method is terminated (step 404).

If that is possible, however, the integrity of the reloaded or recovered protection program is dynamically checked further (step 403).

In an additional variant, provision is made that the computers are connected to a server, the first computer in **Fig. 1**.

The method described above for the protection of a program and/or a file is carried out in each computer.

If the accessing process is not included in the process file, an alarm signal is generated and sent to the server. In the server, in dependency on at least one alarm signal a predetermined action is triggered, for example, a centrally controlled termination of a process.

The following publications were cited within the framework of this document:

- [1] Microsoft Windows NT workstation resource kit: Comprehensive resource guide and utilities for Windows NT 4.0, ISBN 1-57231-343-9, Microsoft Press, C. Doyle (ed.), pp. 117-118, 1996

- [2] Dr. Solomon's Anti-Virus Toolkit Workstation, available to the public on the Internet on July 3, 1998 at Internet address:
<http://www.drsolomon.de/produkte/avtkws/avtkws.html>

- [3] M. Petrek, Under the hood, MS Journal, No. 12, December 1996

- [4] M. Russinovich, Windows NT System-Call Hooking, Dr. Dobb's Journal, pp. 42-46, January 1997

- [5] J. Richter, Advanced Windows, ISBN 1-573231-548-2, 3rd Edition, p. 899 ff., Chapter: Breaking Through Process-Boundary Walls, 1997

- [6] US 5 390 310

- [7] US 5 023 907

ART 34 AMBT

Patent Claims

1. Method for protecting several programs and/or several files from unauthorized access by a process,
 - in which each program and/or each file to be protected is assigned an address space,
 - in which each program and/or each file to be protected is also assigned a process file,
 - in which the process or processes that may run in the address space in question is or are stored in a process file,
 - in which for at least a part of the processes included in a process file, a cryptographic value that uniquely identifies the process is formed,
 - in which the cryptographic value of one process is contained in the process file,
 - in which, during the running of a program and/or a processing of a file to be protected, for a process that attempts to access the address space of the program and/or the file to be protected, a check is made to confirm whether the accessing process is included in the corresponding process file,
 - in which the accessing process's cryptographic value is formed, and
 - in which the cryptographic values of the processes are compared with each other during the check,
 - in which, if the accessing process is included in the process file, the accessing process is started, and
 - otherwise, the accessing process is not started.
2. Method for protecting several programs and/or several files from unauthorized access by a process,
 - in which each program and/or each file to be protected is assigned an address space,
 - in which each program and/or each file to be protected is also assigned a process file,

- in which the process or processes that may run in the address space in question is or are stored in a process file,
- in which for at least a part of the processes included in a process file, a cryptographic value that uniquely identifies the process is formed,
- in which the cryptographic value of one process is contained in the process file,
- in which, during the running of a program and/or a processing of a file to be protected, for a process that attempts to access the address space of the program and/or the file to be protected, a check is made to confirm whether the accessing process is included in the corresponding process file,
- in which the accessing process's cryptographic value is formed, and
- in which the cryptographic values of the processes are compared with each other during the check,
- in which, if the accessing process is included in the process file, the accessing process is continued, and
- otherwise, the accessing process is ended.

3. Method according to one of the Claims 1 or 3 [sic],

in which in a call mechanism for a function of an operating system core with which the programs are executed, a call of the accessing process is forwarded to a checking function in which the check is carried out.

4. Method according to Claim 3,

in which the checking function can be integrated into the address space of the program and/or the file to be protected as a dynamically integrated file.

5. Method according to Claim 1 or 2,

- in which a call of the accessing process is forwarded to a checking function in which the check takes place, and

-in which the checking function is integrated into an operating system core of an operating system with which the programs are executed.

6. Method according to one of the Claims 1 through 5,

in which the operating system is Windows NT.

7. Method according to one of the Claims 1 through 6,

in which at a predetermined interval of time for each active process that runs along with a program and/or a file to be protected, a check is made to confirm whether the active process is contained in the process file that is assigned to the program and/or the file to be protected, and the process is ended if that is not the case.

8. Method according to one of the Claims 1 through 7,

in which in dependency on a predetermined event for each active process that runs along with a program and/or a file to be protected, a check is made to confirm whether the active process is contained in the process file that is assigned to the program to be protected, and the process is ended if that is not the case.

9. Method according to one of the Claims 1 through 8,

in which a protection program for protecting the method according to one of the preceding Claims can be executed is stored encoded and is decoded at the start of the method.

10. Method according to one of the Claims 1 through 9,

in which the programs and/or the files to be protected are stored encoded and are decoded at the

start of the method according to one of the preceding Claims.

11. Method according to Claim 9,

in which after the decoding of the protection program, its integrity is checked and the method according to one of the preceding Claims is executed only if the integrity of the protection program is assured.

12. Method according to Claim 11,

in which after the integrity test of the protection program, the integrity of all processes contained in the process files is checked and the method according to one of the preceding Claims is executed only if the integrity of all of the processes contained in the process files is assured.

13. Method according to Claim 12,

in which after the integrity test of the processes, the integrity of the program and/or the file to be protected is checked and the method according to one of the preceding Claims is executed only if the integrity of the program and/or the file to be protected is assured.

14. Method according to Claim 12 or 13,

in which at least one of the integrity tests takes place through the use of a cryptographic method.

15. Method according to one of the Claims 1 through 14,

in which the cryptographic value is formed through the use of a hash function.

16. Array for protecting several programs from unauthorized access by a process,

with a processor that is set up in a way such that the following steps can be carried out:

- an address space is assigned to each program and/or each file to be protected,
- a process file is assigned to each program and/or each file to be protected,
- the process or processes that may run in the address space in question is or are stored in a process file,
- in which for at least a part of the processes included in a process file, a cryptographic value that uniquely identifies the process is formed,
- in which the cryptographic value of one process is contained in the process file,
- in which, during the running of a program and/or a processing of a file to be protected, for a process that attempts to access the address space of the program and/or the file to be protected, a check is made to confirm whether the accessing process is included in the corresponding process file,
- in which the accessing process's cryptographic value is formed, and
- in which the cryptographic values of the processes are compared with each other during the check,
- in which if the accessing process is included in the process file, the accessing process is started, and
- otherwise, the accessing process is not started.

17. Array for protecting several programs from unauthorized access by a process,
with a processor that is set up in a way such that the following steps can be carried out:

- an address space is assigned to each program and/or each file to be protected,
- a process file is assigned to each program and/or each file to be protected,

- the process or processes that may run in the address space in question is or are stored in a process file,
- in which for at least a part of the processes included in a process file, a cryptographic value that uniquely identifies the process is formed,
- in which the cryptographic value of one process is contained in the process file,
- in which, during the running of a program and/or a processing of a file to be protected, for a process that attempts to access the address space of the program and/or the file to be protected, a check is made to confirm whether the accessing process is included in the corresponding process file,
- in which the accessing process's cryptographic value is formed,
- in which the cryptographic values of the processes are compared with each other during the check,
- in which if the accessing process is included in the process file, the accessing process is continued, and
- otherwise the accessing process is ended.

18. Array according to one of the Claims 16 or 17,

in which the processor is set up in such a way that in a call mechanism for a function of an operating system core with which the programs are executed, a call of the accessing process is forwarded to a checking function in which the check is carried out.

19. Array according to one of the Claims 16 through 18,

in which the processor is set up in such a way that the operating system is Windows NT.

20. Set of several arrays and a server array which is connected with each array of the set of

several arrays and which is to protect several programs from unauthorized access by a process, whereby each array exhibits a processor that is set up in such a way that the following steps can be carried out:

- an address space is assigned to each program and/or each file to be protected,
 - a process file is assigned to each program and/or each file to be protected,
 - the process or processes that may run in the address space in question is or are stored in a process file,
 - in which for at least a part of the processes included in a process file, a cryptographic value that uniquely identifies the process is formed,
 - in which the cryptographic value of one process is contained in the process file,
 - in which, during the running of a program and/or a processing of a file to be protected, for a process that attempts to access the address space of the program and/or the file to be protected, a check is made to confirm whether the accessing process is included in the corresponding process file,
 - in which the accessing process's cryptographic value is formed, and
 - in which the cryptographic values of the processes are compared with each other during the check,
 - in which if the accessing process is included in the process file, the accessing process is started or continued, and
 - otherwise an alarm signal is generated and sent to the server array,
- and whereby the server array exhibits a processor that is set up in such a way that a predetermined action is triggered in dependency on at least one received alarm signal.

Abstract

Method, array and a set of several arrays for protecting several programs and/or several files from unauthorized access by a process

An area and a process file are assigned to each program to be protected. The process or processes that may run in the corresponding area is or are stored in a process file. When the program is running, a process attempting to access the program is checked to confirm whether the accessing process is included in the corresponding process file. The accessing process is executed only if it is included in the process file.

Significant Figure 1

FIG 1

[see drawing]

FIG 2

[see drawing]

Can be reached via API

09/763029

1/6

FIG 1

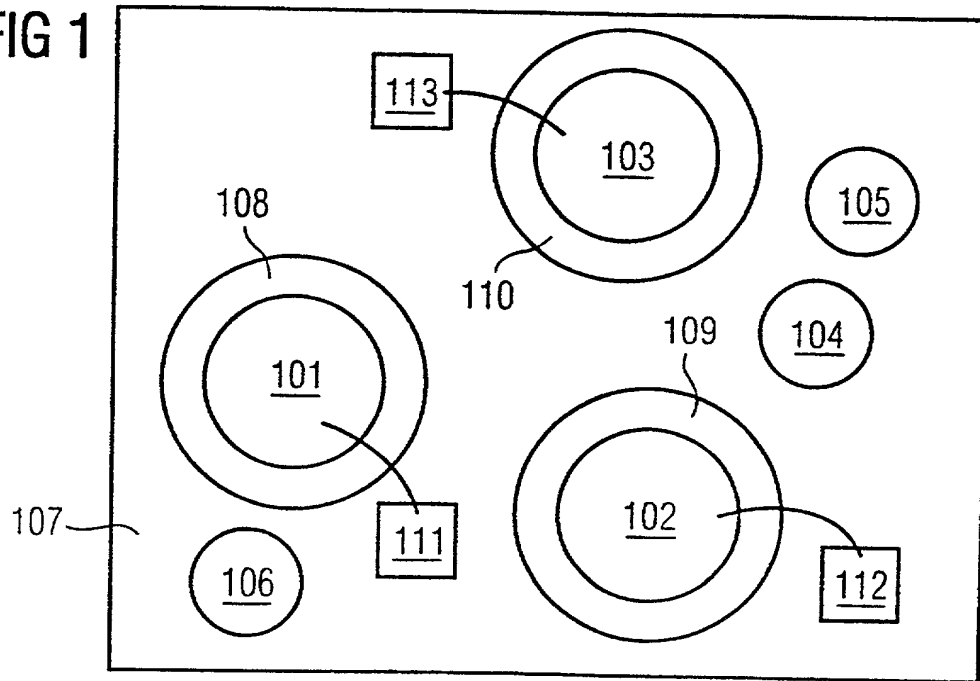


FIG 2

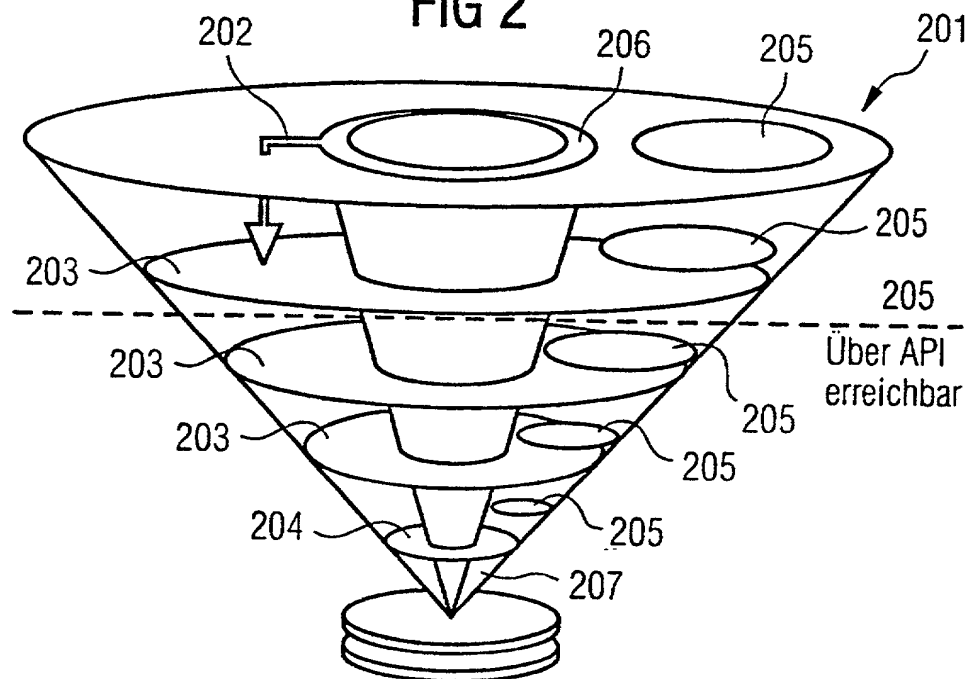


FIG 3

[see drawing]

09 / 763 029

2/6

FIG 3

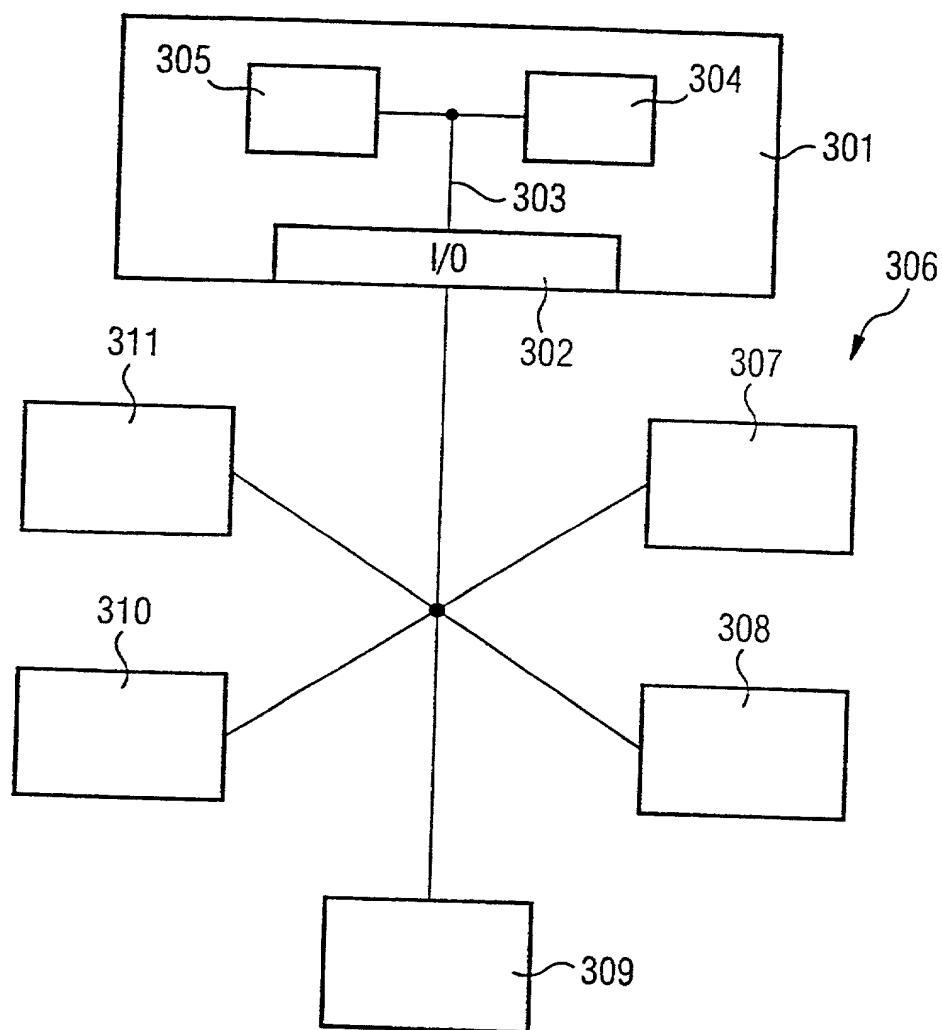


FIG 4

[see flow chart] [note: ja = yes; nein = no]

401 Logon

402 Start SWD concurrent

MONITORING

403 SWD secure? Event or timer?

405 System secure? Event treatment

406 Environment secure?

404 Defined termination, log

407 Start/run application

410 Wait for event

408 Application secure?

409 End?

End

3/6

FIG 4

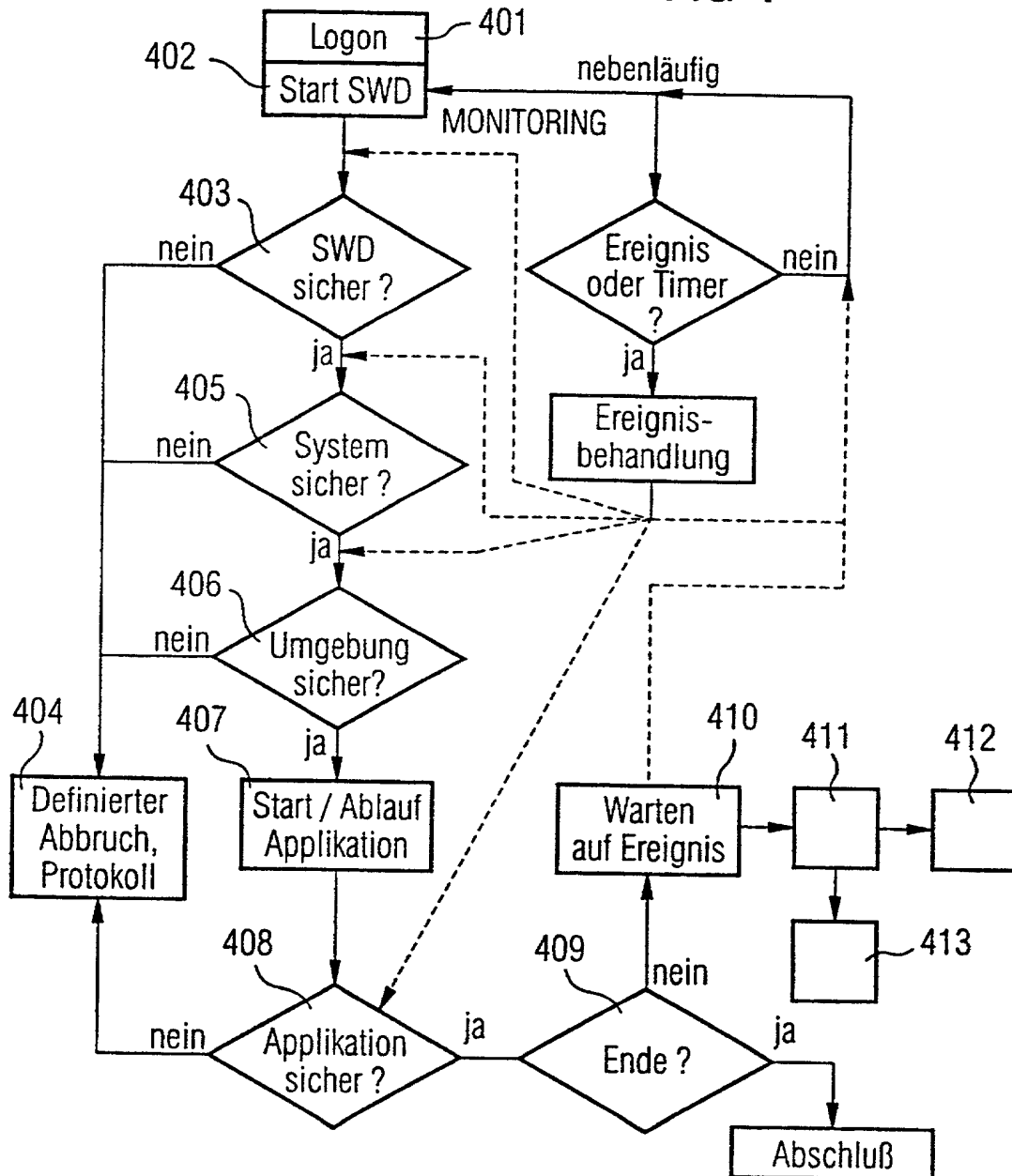
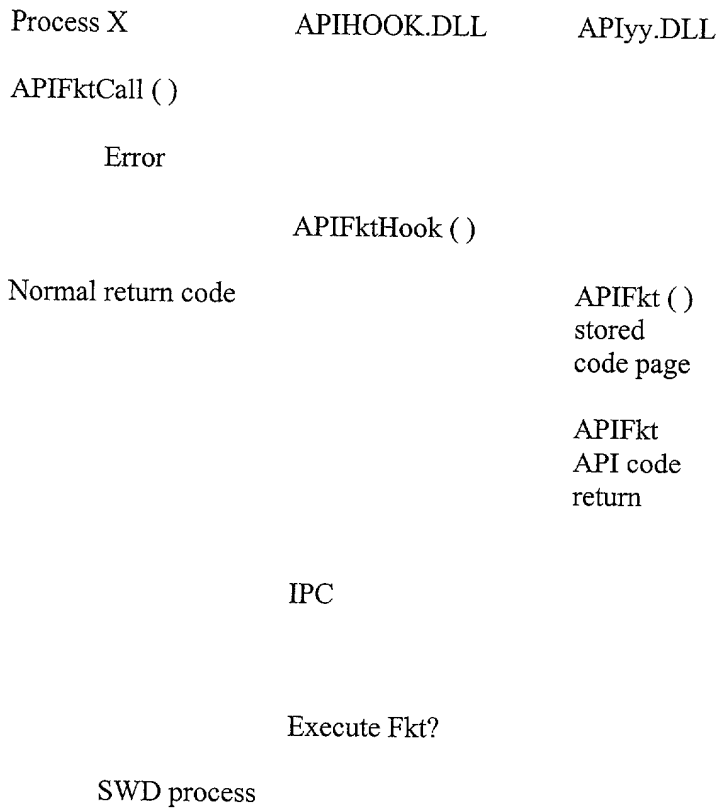


FIG 5

[see figure] [note: ja = yes; nein = no; Fkt = function]



4/6

FIG 5

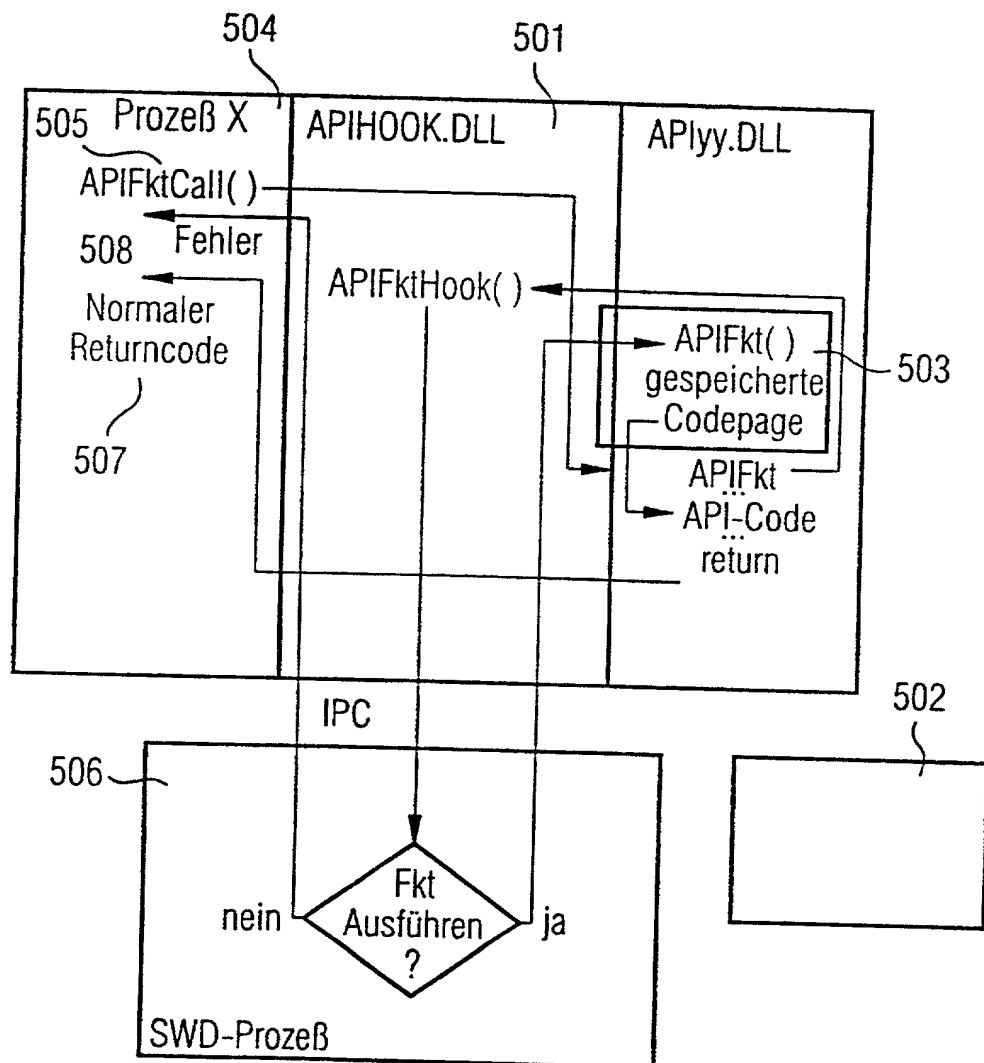
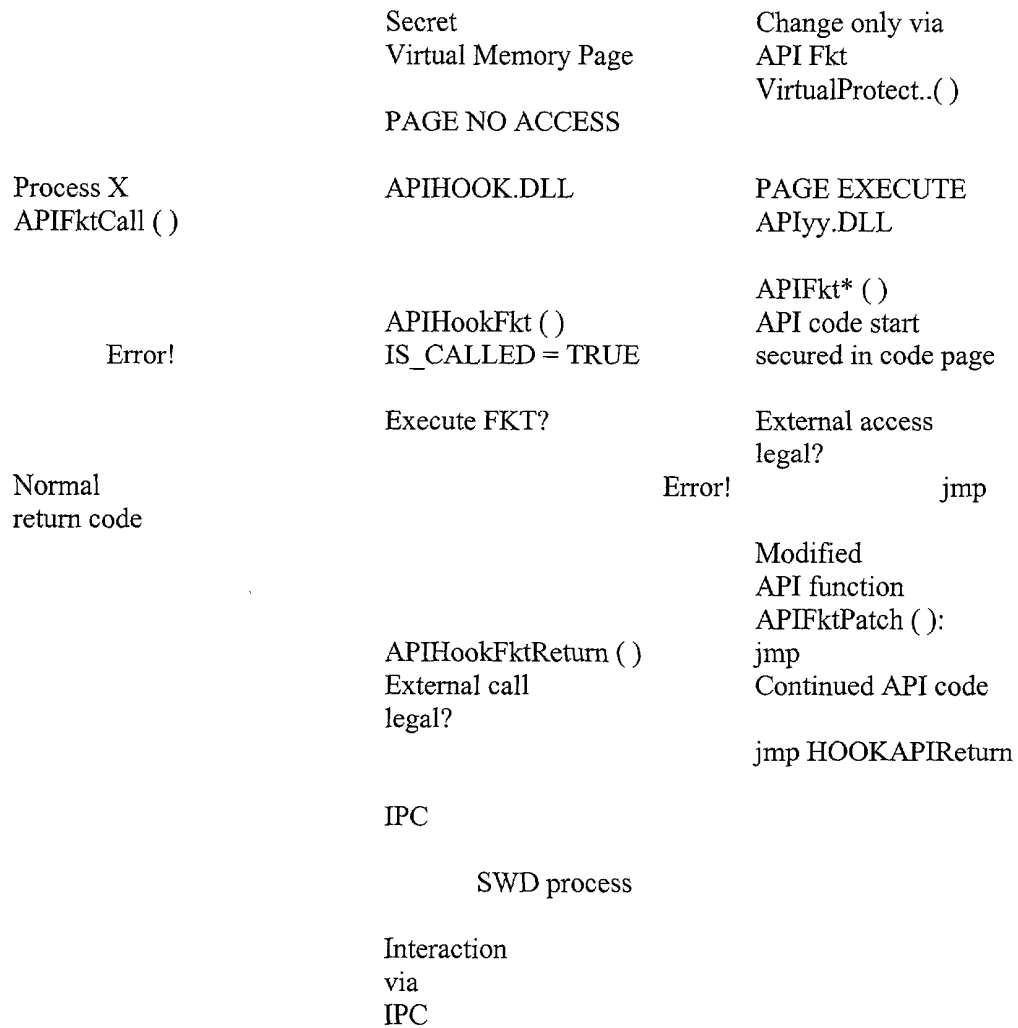


FIG 6

[see figure] [note: ja = yes; nein = no; Fkt = function]



5/6

FIG 6

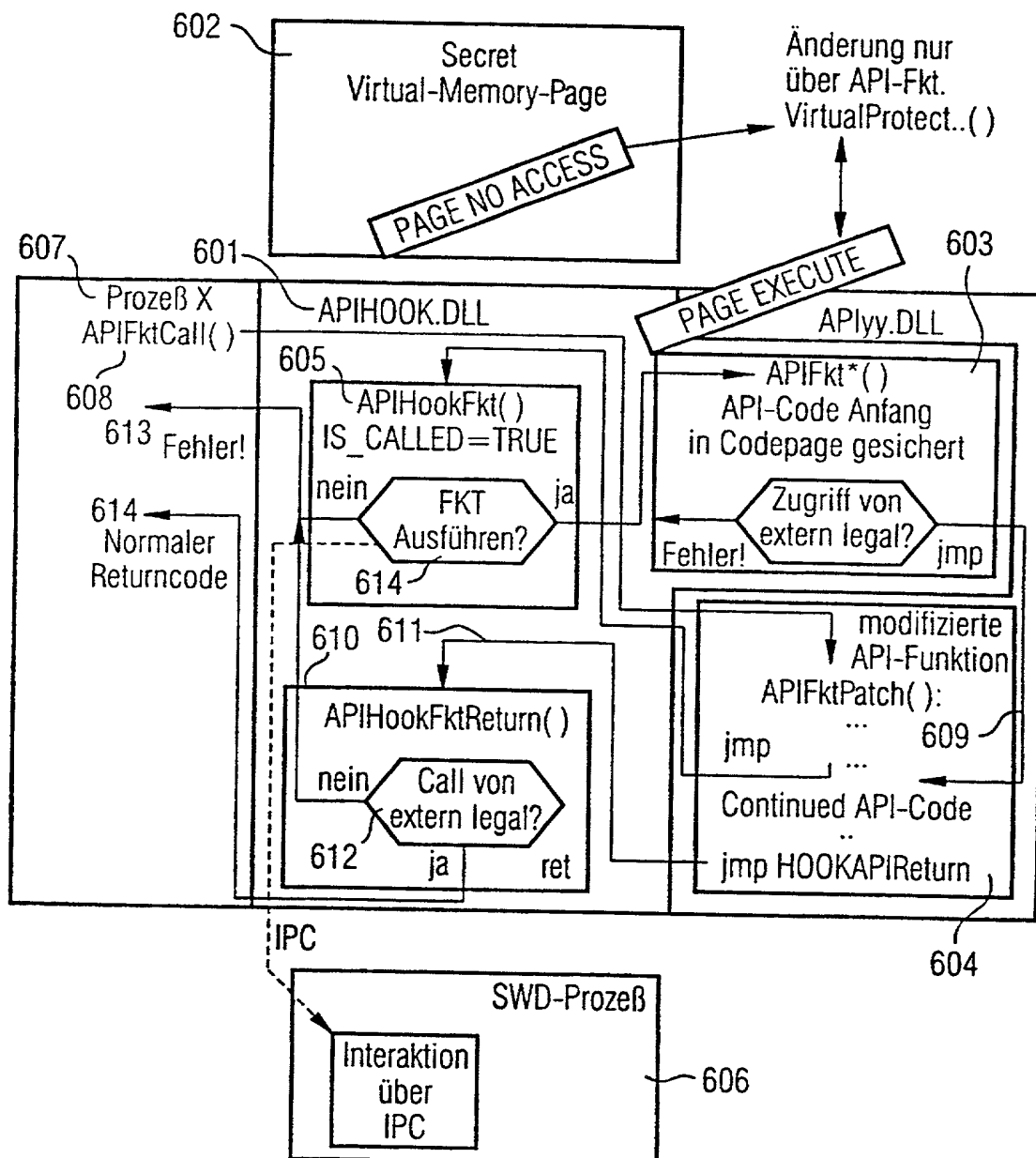
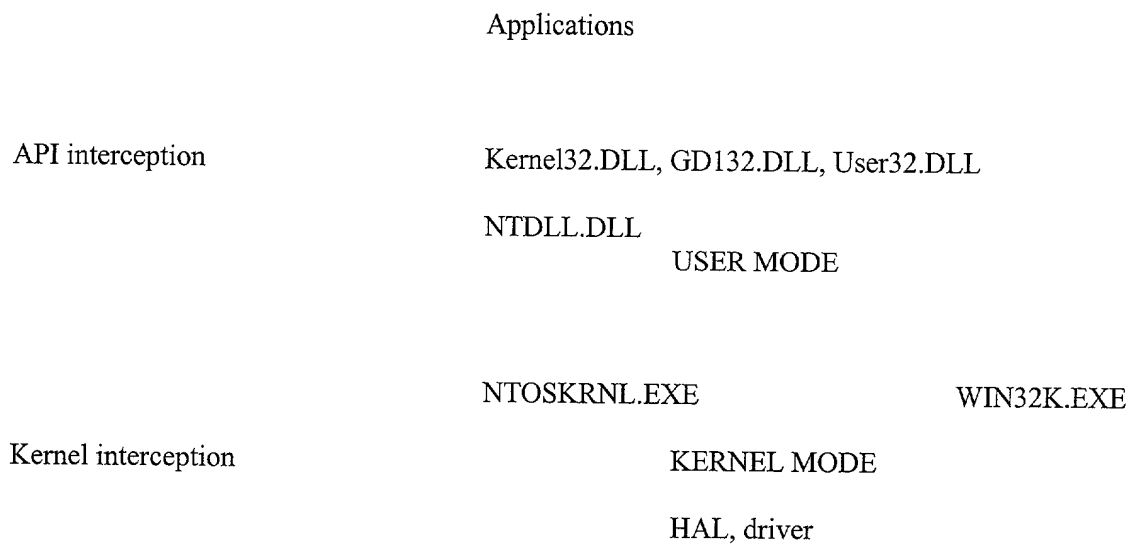
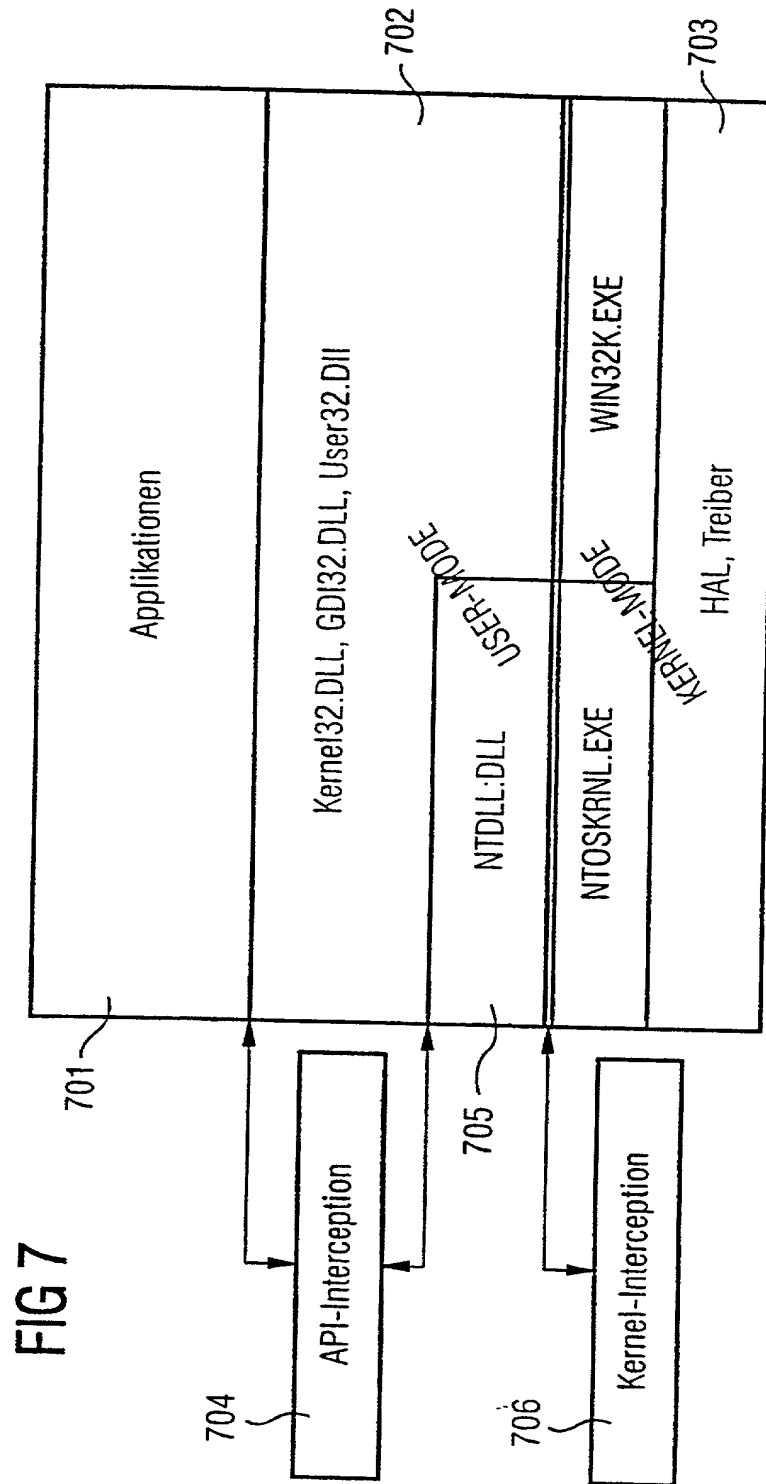


FIG 7

[see figure] [note: ja = yes; nein = no; Fkt = function]



6/6



Declaration and Power of Attorney For Patent Application #3



Erklärung Für Patentanmeldungen Mit Vollmacht

German Language Declaration

Als nachstehend benannter Erfinder erkläre ich hiermit an Eides Statt:

dass mein Wohnsitz, meine Postanschrift, und meine Staatsangehörigkeit den im Nachstehenden nach meinem Namen aufgeführten Angaben entsprechen,

dass ich, nach bestem Wissen der ursprüngliche, erste und alleinige Erfinder (falls nachstehend nur ein Name angegeben ist) oder ein ursprünglicher, erster und Miterfinder (falls nachstehend mehrere Namen aufgeführt sind) des Gegenstandes bin, für den dieser Antrag gestellt wird und für den ein Patent beantragt wird für die Erfindung mit dem Titel:

Verfahren, Anordnung sowie ein Satz
mehrerer Anordnungen zum Schutz
mehrerer Programme und/oder mehrerer
Dateien vor einem unbefugten Zugriff
durch einen Prozess

deren Beschreibung

(zutreffendes ankreuzen)

☐ hier beigefügt ist.

☒ am 01. Juli 1999 als

PCT internationale Anmeldung

PCT Anmeldungsnummer PCT/DE99/02013

eingereicht wurde und am _____
abgeändert wurde (falls tatsächlich abgeändert).

Ich bestätige hiermit, dass ich den Inhalt der obigen Patentanmeldung einschliesslich der Ansprüche durchgesehen und verstanden habe, die eventuell durch einen Zusatzantrag wie oben erwähnt abgeändert wurde.

Ich erkenne meine Pflicht zur Offenbarung irgendwelcher Informationen, die für die Prüfung der vorliegenden Anmeldung in Einklang mit Absatz 37, Bundesgesetzbuch, Paragraph 1.56(a) von Wichtigkeit sind, an.

Ich beanspruche hiermit ausländische Prioritätsvorteile gemäss Abschnitt 35 der Zivilprozessordnung der Vereinigten Staaten, Paragraph 119 aller unten angegebenen Auslandsanmeldungen für ein Patent oder eine Erfindersurkunde, und habe auch alle Auslandsanmeldungen für ein Patent oder eine Erfindersurkunde nachstehend gekennzeichnet, die ein Anmeldedatum haben, das vor dem Anmeldedatum der Anmeldung liegt, für die Priorität beansprucht wird.

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name,

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

the specification of which

(check one)

☐ is attached hereto.

☐ was filed on _____ as

PCT international application

PCT Application No. _____

and was amended on _____
(if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, §1.56(a).

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

German Language Declaration

Prior foreign applications
Priorität beansprucht

Priority Claimed

198 37 666.9 Germany

19. August 1998

(Number) (Country)
(Nummer) (Land)

(Day Month Year Filed)
(Tag Monat Jahr eingereicht)

☒ ☐
Yes No
Ja Nein

(Number) (Country)
(Nummer) (Land)

(Day Month Year Filed)
(Tag Monat Jahr eingereicht)

☐ ☐
Yes No
Ja Nein

(Number) (Country)
(Nummer) (Land)

(Day Month Year Filed)
(Tag Monat Jahr eingereicht)

☐ ☐
Yes No
Ja Nein

Ich beanspruche hiermit gemäss Absatz 35 der Zivilprozessordnung der Vereinigten Staaten, Paragraph 120, den Vorzug aller unten aufgeführten Anmeldungen und falls der Gegenstand aus jedem Anspruch dieser Anmeldung nicht in einer früheren amerikanischen Patentanmeldung laut dem ersten Paragraphen des Absatzes 35 der Zivilprozessordnung der Vereinigten Staaten, Paragraph 122 offenbart ist, erkenne ich gemäss Absatz 37, Bundesgesetzbuch, Paragraph 1.56(a) meine Pflicht zur Offenbarung von Informationen an, die zwischen dem Anmeldedatum der früheren Anmeldung und dem nationalen oder PCT internationalen Anmeldedatum dieser Anmeldung bekannt geworden sind.

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §122, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application.

(Application Serial No.)
(Anmeldeseriennummer)

(Filing Date)
(Anmeldedatum)

(Status)
(patentiert, anhängig,
aufgegeben)

(Status)
(patented, pending,
abandoned)

(Application Serial No.)
(Anmeldeseriennummer)

(Filing Date)
(Anmeldedatum)

(Status)
(patentiert, anhängig,
aufgeben)

(Status)
(patented, pending,
abandoned)

Ich erkläre hiermit, dass alle von mir in der vorliegenden Erklärung gemachten Angaben nach meinem besten Wissen und Gewissen der vollen Wahrheit entsprechen, und dass ich diese eidesstattliche Erklärung in Kenntnis dessen abgebe, dass wissentlich und vorsätzlich falsche Angaben gemäss Paragraph 1001, Absatz 18 der Zivilprozessordnung der Vereinigten Staaten von Amerika mit Geldstrafe belegt und/oder Gefängnis bestraft werden können, und dass derartig wissentlich und vorsätzlich falsche Angaben die Gültigkeit der vorliegenden Patentanmeldung oder eines darauf erteilten Patentes gefährden können.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

German Language Declaration

VERTRETUNGSVOLLMACHT: Als benannter Erfinder beauftrage ich hiermit den nachstehend benannten Patentanwalt (oder die nachstehend benannten Patentanwälte) und/oder Patent-Agenten mit der Verfolgung der vorliegenden Patentanmeldung sowie mit der Abwicklung aller damit verbundenen Geschäfte vor dem Patent- und Warenzeichenamt: (Name und Registrationsnummer anführen)

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith. (list name and registration number)

Messrs.

And I hereby appoint

James D. Halsey, Jr. (Reg. No. 22,729); Harry John Staas (Reg. No. 22,010); David M. Pitcher (Reg. No. 25,908); John C. Garvey (Reg. No. 28,607); J. Randall Beckers (Reg. No. 30,358); William F. Herbert (Reg. No. 31,024); Richard A. Gollhofer (Reg. No. 31,106); Mark J. Henry (Reg. No. 36,162); Paul I. Kravetz (Reg. No. 35,230); Gene M. Garner II (Reg. No. 34,172); Michael D. Stein (Reg. No. 37,240); Todd E. Marlette (Reg. No. 35,269); Norman L. Ourada (Reg. No. 41,235); Deborah S. Gladstein (Reg. No. 43,636); Jon H. Muskin (Reg. No. 43,824); Stephen Boughner (Reg. No. 45,317); John H. Stowe (Reg. No. 32,863); C. Joan Gilsdorf (Reg. No. 43,635); Mehdi Sheikerz (Reg. No. 41,307); James G. McEwen (Reg. No. 41,983); Michael J. Badagliacca (Reg. No. 39,099); Alicia M. Choi (Reg. No. P-46,621); Jon F. Hadidi (Reg. No. 46,427); and William M. Schertler (Reg. No. 35,348 (agent)).

Telefongespräche bitte richten an:
(Name und Telefonnummer)

Direct Telephone Calls to: (name and telephone number)

(202) 434-1500
Ext. _____

Postanschrift:

Send Correspondence to:

Staas & Halsey LLP
700 Eleventh Street, N.W.
Washington, D.C. 20001
U.S.A.
Customer No. 21171

Voller Name des einzigen oder ursprünglichen Erfinders:		Full name of sole or first inventor:	
SCHÄFER, Manfred			
Unterschrift des Erfinders <i>Manfred Schäfer</i>	Datum <i>19.2.01</i>	Inventor's signature	Date
Wohnsitz D-85661 Forstinning, Germany		Residence <i>DEX</i>	
Staatsangehörigkeit Bundesrepublik Deutschland		Citizenship	
Postanschrift St.-Josef-Str. 16		Post Office Address	
D-85661 Forstinning			
Bundesrepublik Deutschland			
Voller Name des zweiten Miterfinders (falls zutreffend):		Full name of second joint inventor, if any:	
Unterschrift des Erfinders	Datum	Second inventor's signature	Date
Wohnsitz		Residence	
Staatsangehörigkeit		Citizenship	
Postanschrift		Post Office Address	

(Bitte entsprechende Informationen und Unterschriften im Falle von dritten und weiteren Miterfindern angeben).

(Supply similar information and signature for third and subsequent joint inventors).